Scalable Optimization in the Cloud with GAMS and GAMS Engine

Stefan Mann, Frederik Proske, Michael Bussieck

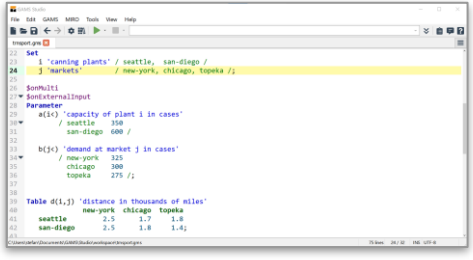



Hamdi Burak Usul

GAMS

Modeling Platform

- **Platform Independent Algebraic Modeling Language for Domain Experts**
- **Commercial and Academic Solvers** (packaged)
- **APIs for major programming languages** (C++, Java, Python, Matlab and more)

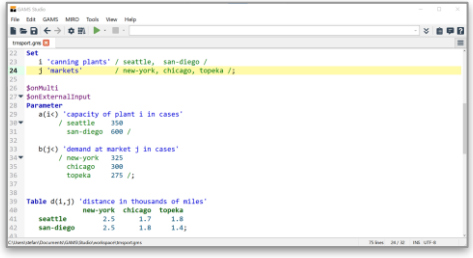


GAMS

16:30 - 18:00 Modeling languages and platforms

Modeling Platform

- **Platform Independent Algebraic Modeling Language for Domain Experts**
- **Commercial and Academic Solvers** (packaged)
- **APIs for major programming languages** (C++, Java, Python, Matlab and more)



MIRO

Graphical UI Generator

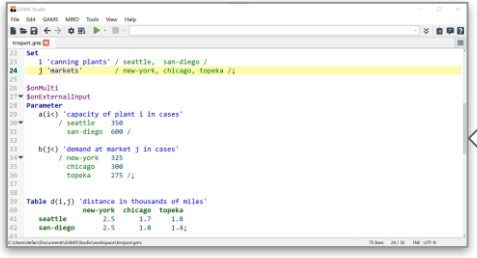
- **Turns models into web applications for end users**
- **Extendable with custom code**
- **Local or Server Installation**

GAMS

16:30 - 18:00 Modeling languages and platforms

Modeling Platform

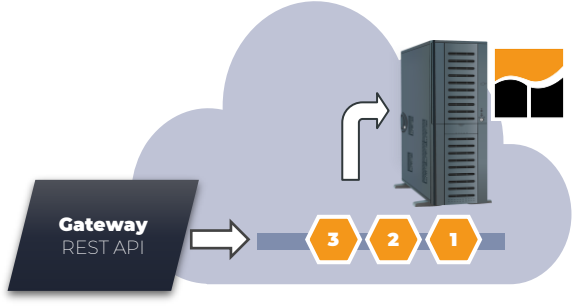
- **Platform Independent Algebraic Modeling Language for Domain Experts**
- **Commercial and Academic Solvers** (packaged)
- **APIs for major programming languages** (C++, Java, Python, Matlab and more)



MIRO

Graphical UI Generator

- **Turns models into web applications for end users**
- **Extendable with custom code**
- **Local or Server Installation**

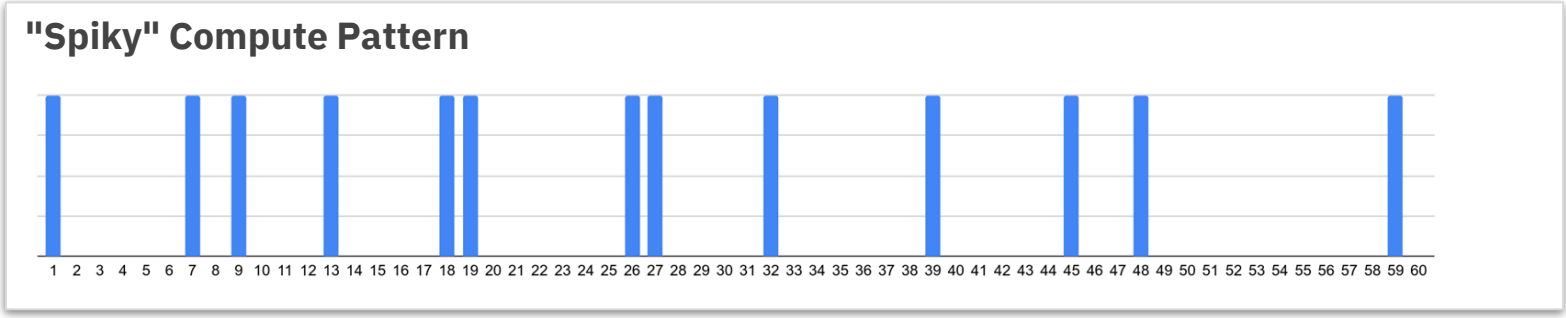


ENGINE

Deployment Solution

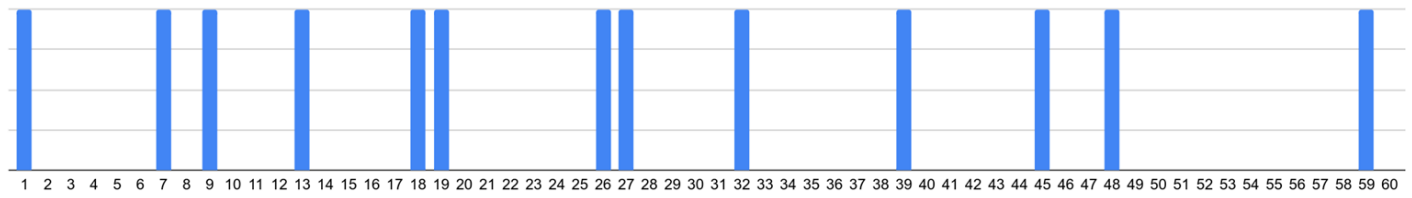
- **Solves GAMS models on centralized resources** (on-prem or cloud)
- **REST API** (user & job management)
- **GAMS job scheduling & Load balancing**

Motivation

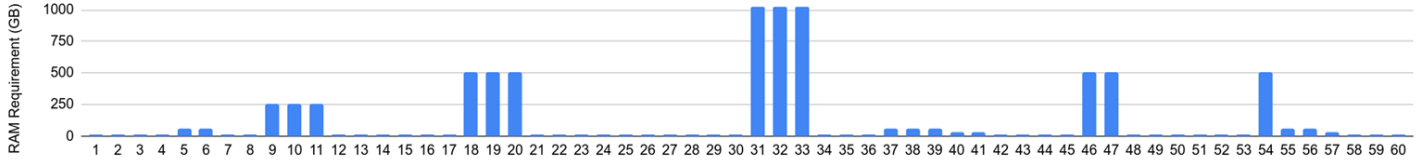

"Spiky" Compute Pattern
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60

Motivation

"Spiky" Compute Pattern

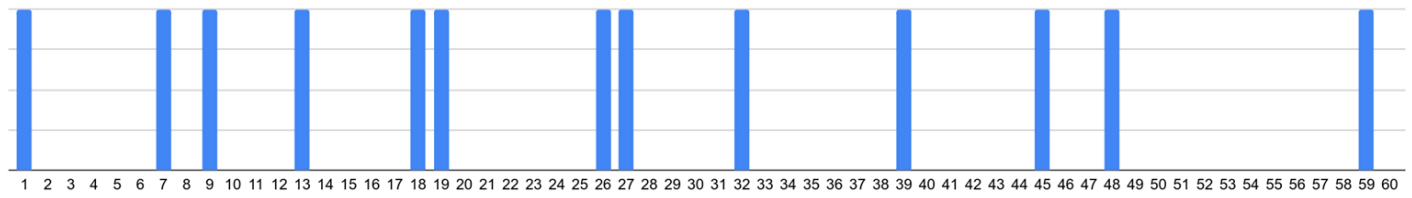


Fluctuating Resource Requirements

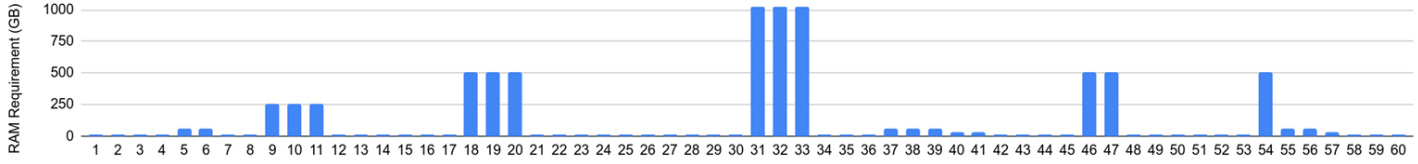


Motivation

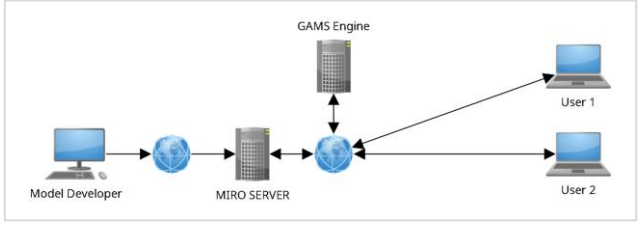
"Spiky" Compute Pattern

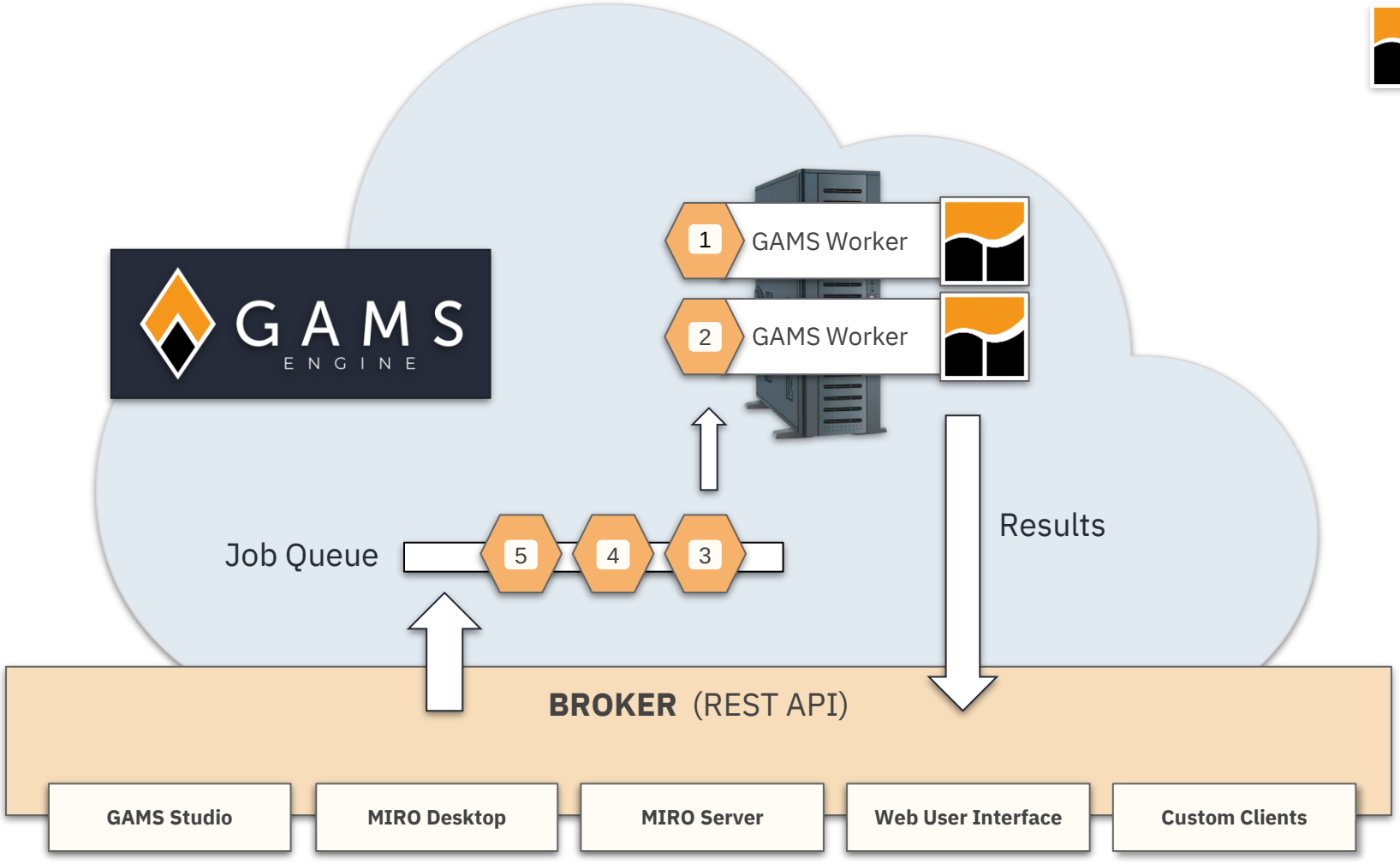


Fluctuating Resource Requirements

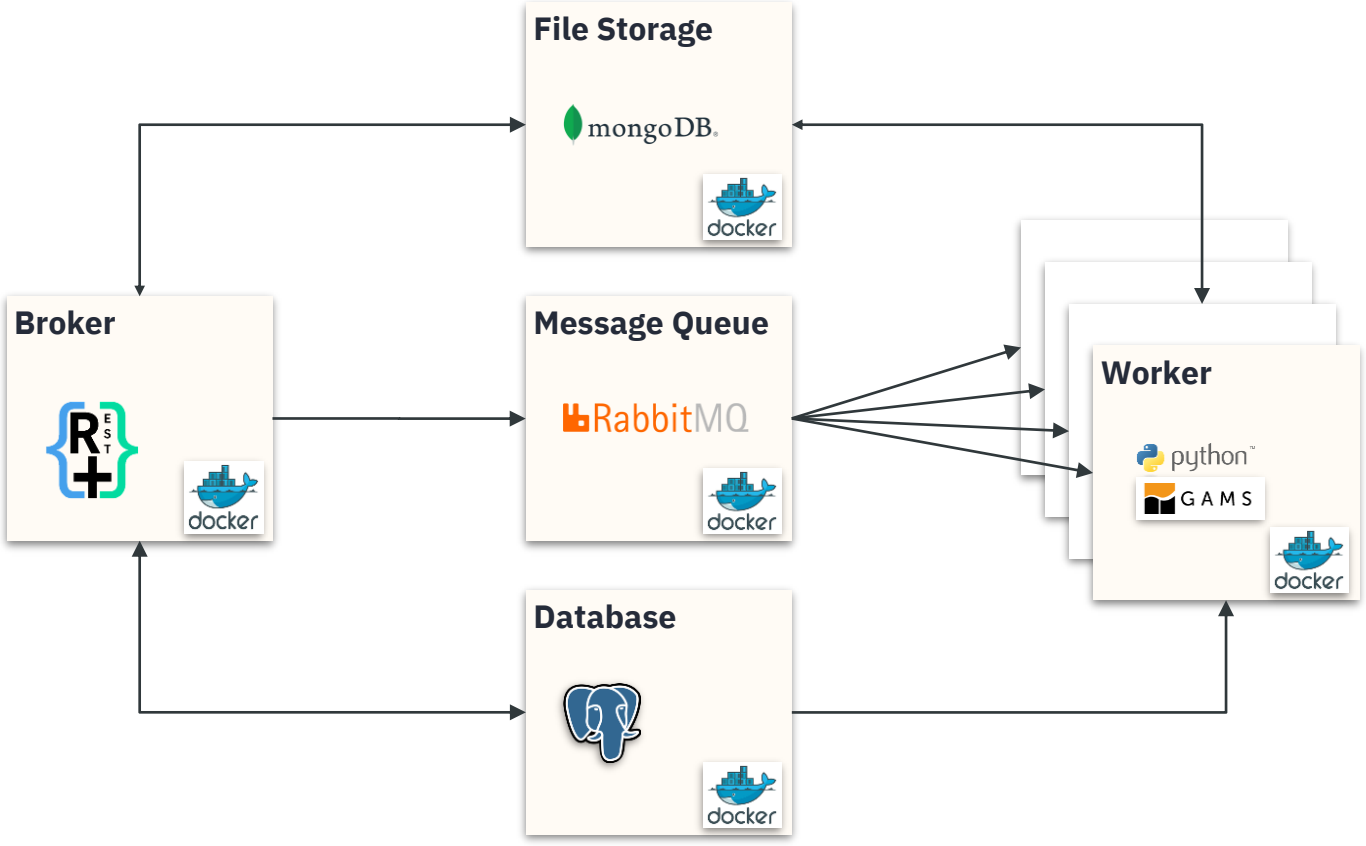


Model Sharing

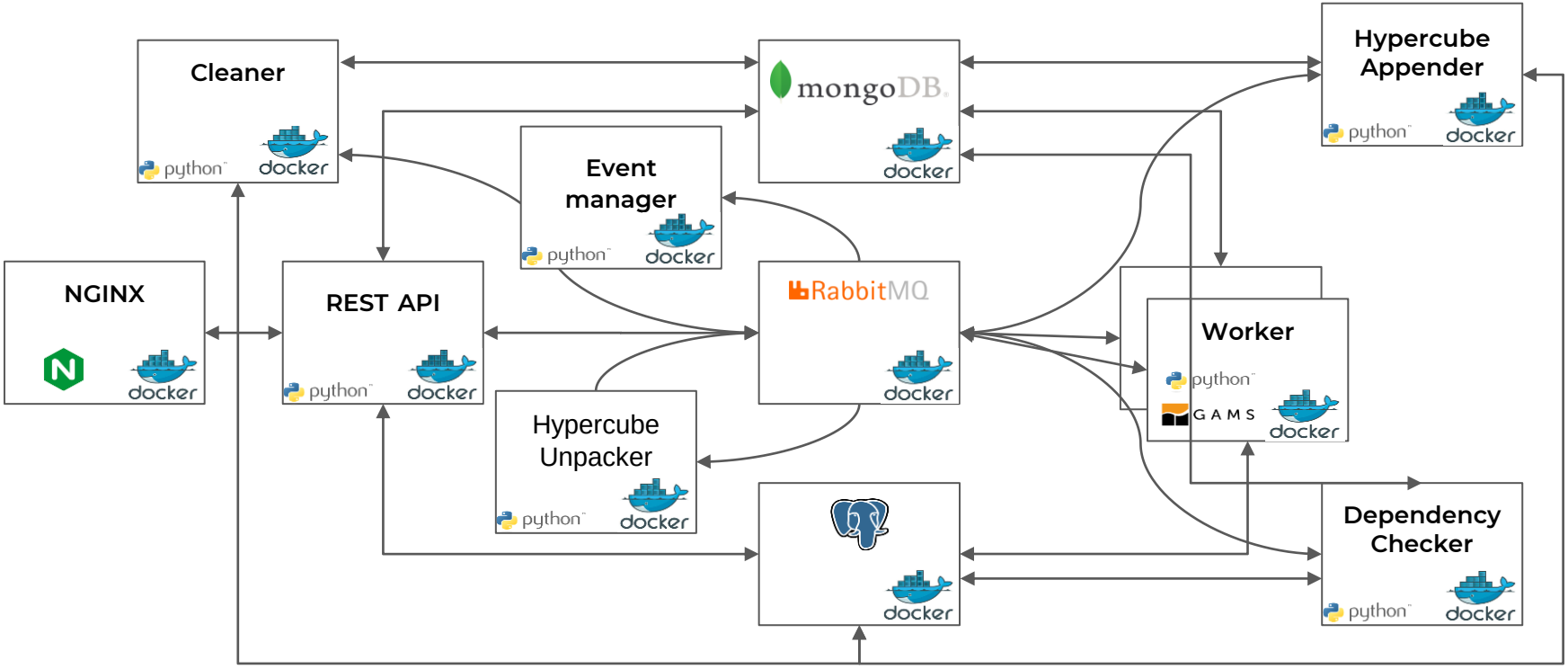



GAMS
ENGINE
1
GAMS Worker
2
GAMS Worker
Job Queue
5
4
3
Results
BROKER (REST API)
GAMS Studio
MIRO Desktop
MIRO Server
Web User Interface
Custom Clients

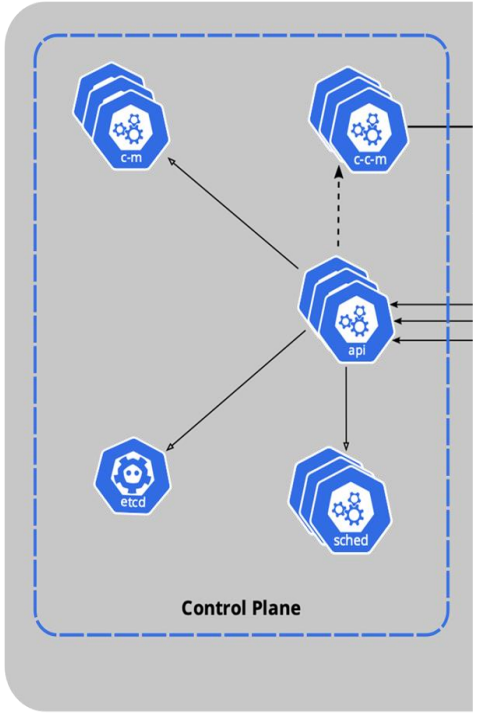
GAMS Engine (simplified) Docker stack



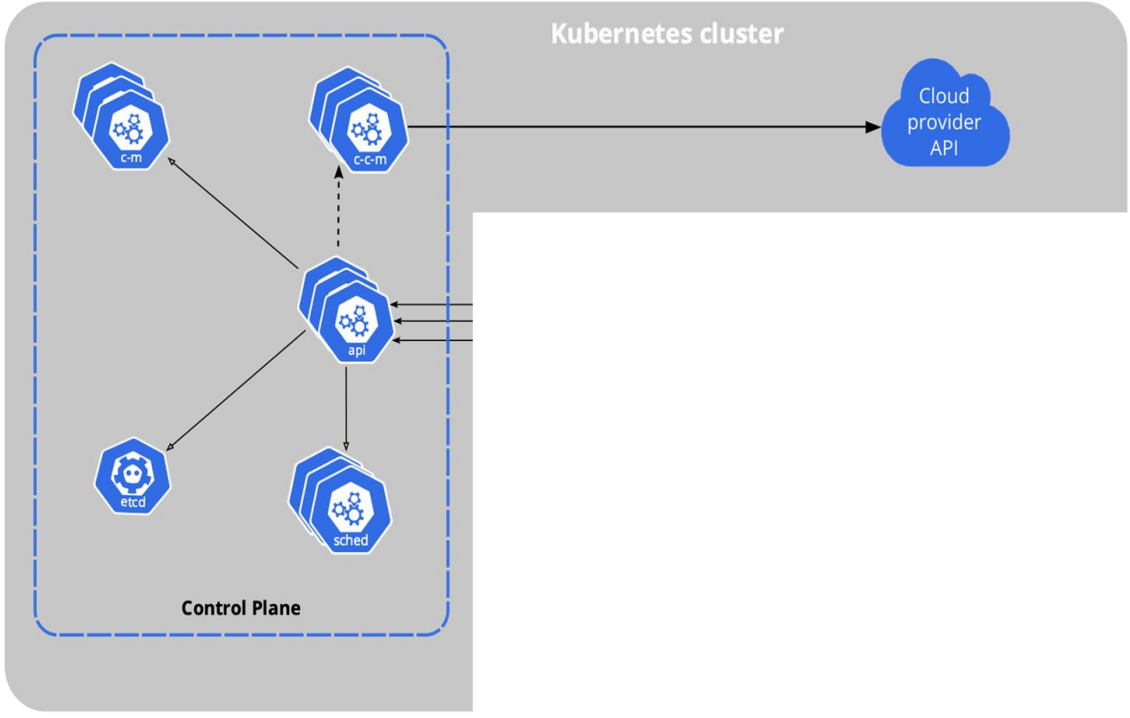
GAMS Engine Docker stack



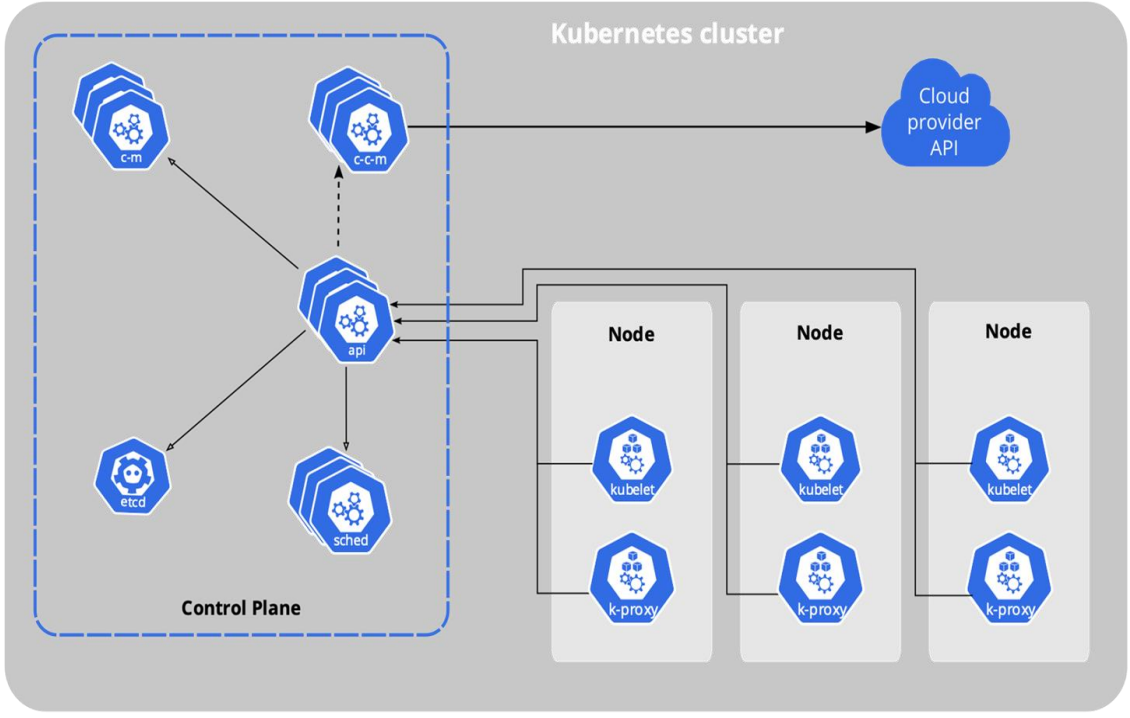
Orchestration: Kubernetes

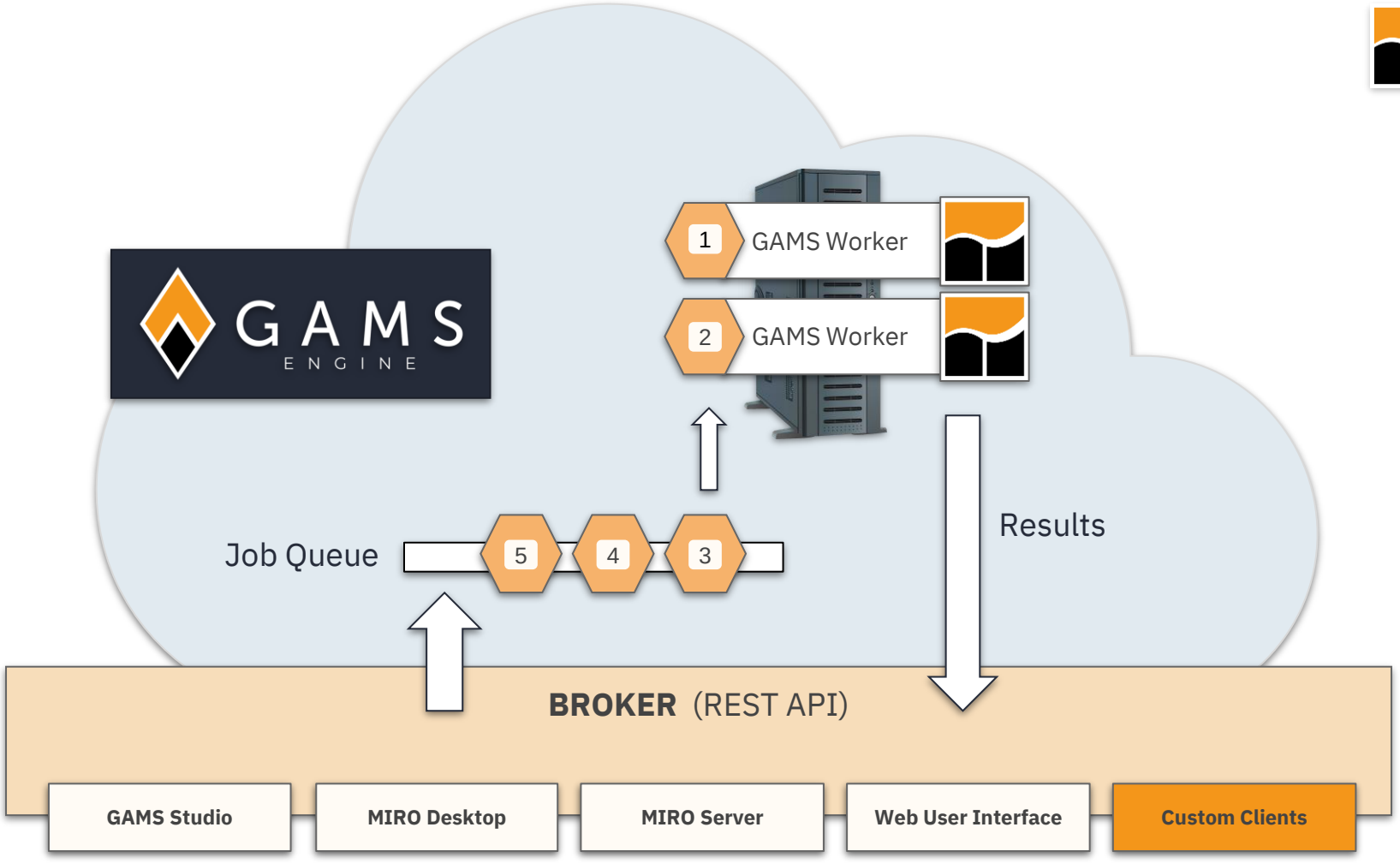


Orchestration: Kubernetes



Orchestration: Kubernetes




GAMS
ENGINE
1
GAMS Worker
2
GAMS Worker
Job Queue
5
4
3
Results
BROKER (REST API)
GAMS Studio
MIRO Desktop
MIRO Server
Web User Interface
Custom Clients

Authentication (GAMS Engine SaaS)

In [33]:
```
import requests
from requests.auth import HTTPBasicAuth
import time


au = HTTPBasicAuth("john_doe", password)
url = "https://engine.gams.com/api"
```

First Request

In [2]:
```
r = requests.get(url + '/usage/instances/john_doe', auth=au)
r.json()
```

Out[2]:
```
{'instances_inherited_from': 'john_doe',
 'default_inherited_from': 'john_doe',
 'instances_available': [{'label': 'TIMES_z1d.large',
   'cpu_request': 1.8,
   'memory_request': 15070,
   'workspace_request': 50000,
   'node_selectors': [{'key': 'gams.com/instanceType', 'value': 'z1d.large'}],
   'tolerations': [],
   'multiplier': 1.0},
  {'label': 'GAMS_z1d.xlarge_A',
   'cpu_request': 3.8,
   'memory_request': 30710,
   'workspace_request': 50000,
   'node_selectors': [{'key': 'gams.com/instanceType', 'value': 'z1d.xlarge'}],
   'tolerations': [],
   'multiplier': 1.1}],
 'default_instance': {'label': 'TIMES_z1d.large',
  'cpu_request': 1.8,
  'memory_request': 15070,
  'workspace_request': 50000,
  'node_selectors': [{'key': 'gams.com/instanceType', 'value': 'z1d.large'}],
  'tolerations': [],
  'multiplier': 1.0}}
```

Submitting a Simple Job

In [5]:
```
from zipfile import ZipFile

with ZipFile('model.zip','w') as zip:
            zip.write('trnsport.gms')

query_params = {
    'model': 'trnsport',
    'namespace': 'demo',
    'labels': 'instance=TIMES_z1d.large'
}

# Create dict with model zip file
job_files = {'model_data': open('model.zip','rb')}


r = requests.post(url + '/jobs/', params=query_params, files=job_files, auth=au)

token =  r.json()['token']
print(token)
```

```
bf6c3474-a853-49c5-acdd-3d11b11505fa
```

Submitting a Simple Job

In [5]:
```
from zipfile import ZipFile

with ZipFile('model.zip','w') as zip:
            zip.write('trnsport.gms')

query_params = {
    'model': 'trnsport',
    'namespace': 'demo',
    'labels': 'instance=TIMES_z1d.large'
}

# Create dict with model zip file
job_files = {'model_data': open('model.zip','rb')}


r = requests.post(url + '/jobs/', params=query_params, files=job_files, auth=au)

token =  r.json()['token']
print(token)
```

Choosing the hardware to use!

```
bf6c3474-a853-49c5-acdd-3d11b11505fa
```

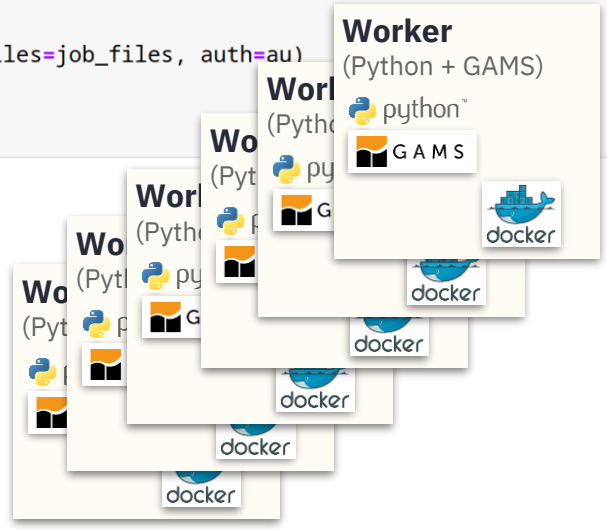
Submitting a Simple Job

In [5]:
```
from zipfile import ZipFile

with ZipFile('model.zip','w') as zip:
            zip.write('trnsport.gms')

query_params = {
    'model': 'trnsport',
    'namespace': 'demo',
    'labels': 'instance=TIMES_z1d.large'
}

# Create dict with model zip file
job_files = {'model_data': open('model.zip','rb')}


r = requests.post(url + '/jobs/', params=query_params, files=job_files, auth=au)

token =  r.json()['token']
print(token)
```

```
bf6c3474-a853-49c5-acdd-3d11b11505fa
```

Choosing the hardware to use!

Autoscaling!



Job Status

In [41]:
```
r = requests.get(url + '/jobs/' + token, auth=au)
r.json()
```

Out[41]:
```
{'token': '3f0ad369-1cb7-4430-9a25-afc323688b72',
 'model': 'trnsport',
 'is_temporary_model': True,
 'is_data_provided': False,
 'status': 10,
 'process_status': 0,
 'stdout_filename': 'log_stdout.txt',
 'namespace': 'demo',
 'stream_entries': [],
 'arguments': [],
 'submitted_at': '2022-03-04T09:10:55.742819+00:00',
 'finished_at': '2022-03-04T09:11:00.287470+00:00',
 'user': {'username': 'john_doe', 'deleted': False, 'old_username': None},
 'text_entries': [],
 'dep_tokens': [],
 'labels': {'cpu_request': 1.8,
  'memory_request': 15070,
  'workspace_request': 50000,
  'tolerations': [],
  'node_selectors': [{'key': 'gams.com/instanceType', 'value': 'z1d.large'}]},
 'result_exists': True}
```

Downloading Job Results

In [42]:
```
r = requests.get(url + '/jobs/' + token + '/result', auth=au)

file = open('results.zip','wb')
file.write(r.content)
file.close()
```

By default, the zip file contains the GAMS log for the run, a copy of the model file, and the lst file. Here is a section of the log that shows we did indeed successfully solve the model on GAMS Engine:

```
   Iteration      Dual Objective            In Variable           Out Variable
        1             73.125000    x(seattle,new-york) demand(new-york) slack
        2            119.025000     x(seattle,chicago)  demand(chicago) slack
        3            153.675000    x(san-diego,topeka)   demand(topeka) slack
        4            153.675000  x(san-diego,new-york)  supply(seattle) slack

   --- LP status (1): optimal.
   --- Cplex Time: 0.11sec (det. 0.01 ticks)


   Optimal solution found
   Objective:          153.675000
```

Features not shown

- MIRO and Studio integration
- model registration
- job dependencies
- powerful user and access management

Around the corner

- worker pools
- single sign-on with OAUTH2 and LDAP

Contact Us

sales@gams.com
www.gams.com

@GamsSoftware

https://www.linkedin.com/company/gams-development